We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

27

Acquisition of Obstacle Avoidance Actions with Free-Gait for Quadruped Robots

Tomohiro Yamaguchi*, Keigo Watanabe** and Kiyotaka Izumi**
** Department of Electrical, Electronics and Information Engineering, Kanagawa University*
*** Department of Advanced Systems Control Engineering, Saga University*
Japan

1. Introduction

Although legged mobile robots are inferior to wheeled or crawler types in mobility efficiency on the flat ground, they demonstrate high motion performance and adaptation capability to the ground by utilizing their high degrees of freedom (DOF). Since such robots choose stable leg-placement, stable movements can be performed on irregular terrains (Şafak & Adams, 2002). Moreover, they demonstrate some unique functionalities, e.g., the scaffold of a stable activity at the time of rest can be formulated from taking the posture of legs which are hard to topple (Hirose & Yoneda, 1993).

However, it is very difficult to decide robot gait due to its high DOF. When the legs of the robot are simply controlled by a fixed command, adaptation capability to the terrain is remarkably restricted and sometimes it is impossible to maintain a stable walk. Moreover, when a leg is unable to be placed properly an optimum leg placement must be efficiently found from among other candidates.

Therefore, it needs for a legged mobile robot to sequentially decide the progression of legs. For that purpose, the robot predictively perceives and recognizes geographical features of the terrain, and it consequently gets over any obstacle by using an adaptive ability acquired in advance. From this fact, legged robots are not necessary to avoid all the obstacles by altering their path, unlike wheeled or crawler types, because they can avoid an obstacle by crawling-over or striding, according to the obstacle's nature and the current state of the robot. Thus, it can be found that the mobility efficiency to reach a destination is improved by such action. Moreover, when robots have many legs like 4-legged or 6-legged types, the movement range is affected by the order of swing leg.

We studied path to a destination and obstacle avoidance of a quadruped robot considering free-gait. In general, quadruped robots can realize two types of walk: one is the static walk which retains the stability statically, keeping the center of gravity (COG) in the polygon constructed by the support legs, and the other is the dynamic walk which retains dynamic stability, though it is statically unstable. In the static walk, one feature is that an irregular-terrain walk is easily realized, whereas excelling in walking-speed or consumption energy is a feature of the dynamic walk (Kimura et al., 1990, Kimura, 1993). Thus, static walk is suitable for action acquisition of quadruped robots in geographical environments where obstacles, such as a level difference, exist (Chen et al., 2002, Chen et al., 2002).

Source: Mobile Robots, Moving Intelligence, ISBN: 3-86611-284-X, Edited by Jonas Buchli, pp. 576, ARS/plV, Germany, December 2006

In this research, since obstacle avoidance is taken into consideration, the static walk is adopted as a basic walk and the order of swing leg is determined. A free gait planning in the static walk was already formulated as a conditional optimization problem and the solution method by the Monte Carlo method was proposed by Nakamura et al. (Nakamura et al., 1999). However, assumed, fixed environment specializes the result obtained from this method; it has no flexibilities to the outside of search environment.

Dimensions of the obstacle and the current state of the robot can be perceived accurately, because of the presence of force sensors, ultrasonic sensors and potentiometers on the present quadruped robot. A simulator is built based on the robot's structure and the path to the destination is acquired from the simulator. When avoiding an obstacle by quadruped robot, there are many combinatorial solutions, such as combinations of turn and forward movements, combinations of sideway and forward movements, etc. The robot's body height must be regulated because leg movement is restricted by height. When the leg is placed on a corner of an obstacle, the robot may fall, so robot action must be determined from the relative position between the obstacle and the robot. We propose a method for determining the action of a quadruped robot using neural network (NN) from the position of the destination, the obstacle configuration, and the robot's self-state. Note, however, that no any free-gait motion is taken into consideration at the first research. The order of swing leg in free-gait is determined in the second research using the amount of movements and the robot's self-state. The static walk of the quadruped robot has 24 kinds of the order of swing leg. Since the static walk always needs to set the COG of the robot in the polygon constructed by the supporting legs, the amount of movements of the body is different, depending on the order of swing leg. Therefore, the order of swing leg is determined by another NN. Furthermore, an NN for determining the robot action is acquired by re-learning the NN that was built in the case when the order of swing leg was fixed. To reach a destination with a minimum number of walking cycles (Furusho, 1993), NN design parameters are optimized by a genetic algorithm (GA) using data from several environments, in which each environment has different destinations and obstacle dimensions.

2. Quadruped Robot

Fig. 1 shows the experimental setup. TITAN-VIII (Hirose & Arikawa 1999) (see Fig. 2) is the quadruped robot. TITAN-VIII has four legs, one with three DOF, and each joint has a potentiometer. Force sensing resistors (Interlink Electronics, FSR Part # 402) are used on the leg sole to measure force exerted on each leg. Ultrasonic sensors (NiceRa, T/R40-16) are used on the forelegs to detect an obstacle; each foreleg has three ultrasonic sensors.

Potentiometer measurement and force sensing resistor are transferred to a personal computer through a robot interface board (Fujitsu, RIF-01) and an A/D converter board (Interface Corporation, PCI-3133). The ultrasonic sensor measures the time difference between emittance and reception of ultrasonic waves, which is reflected on an obstacle by universal pulse processor (UPP), part of RIF-01. The computer sends joint angle commands to a motor driver (Okazaki Sangyo Co. Ltd., Titech Motor Driver) on the robot through the robot interface board. Since sensor information in feedback control must be processed in real time, RT-Linux is used as the computer OS.

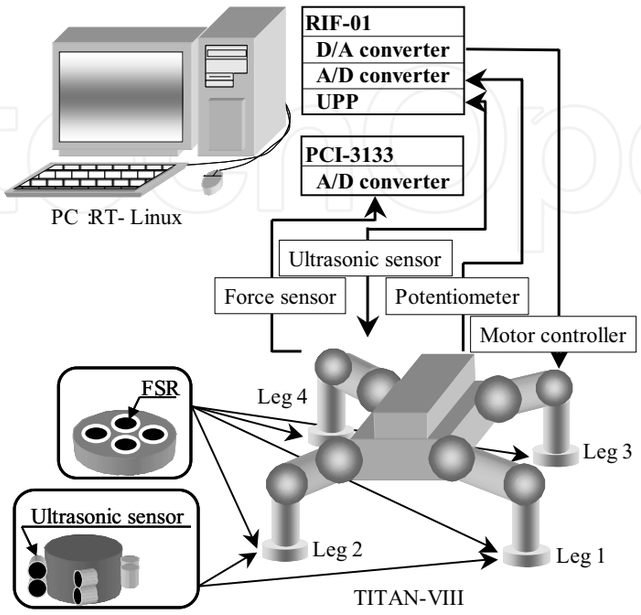

Fig. 1. Robot control system.

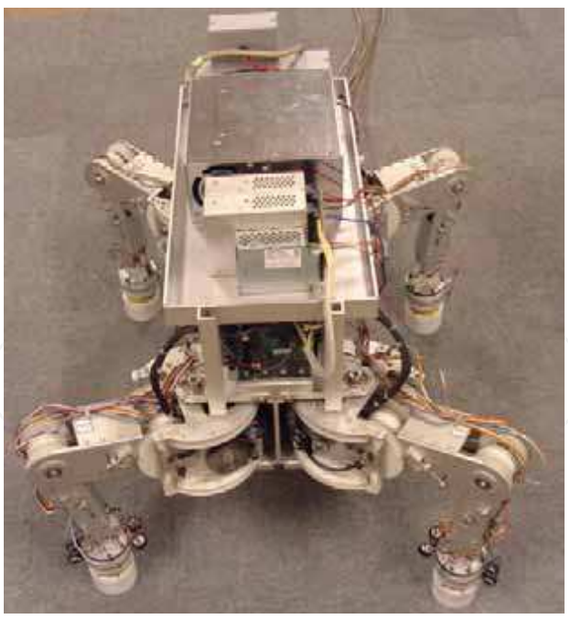

Fig. 2. TITAN-VIII.

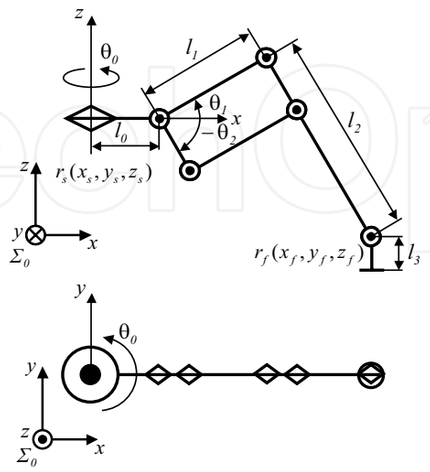

Fig. 3. Coordinate system of TITAN-VIII's leg.

The coordinate system of one leg of the robot is defined in Fig. 3. For any leg, assuming position coordinates of a shoulder are $r_{si}(x_{si}, y_{si}, z_{si})$ and position coordinates of a sole are $r_{fi}(x_{fi}, y_{fi}, z_{fi})$, joint angles, $\theta(\theta_{0i}, \theta_{1i}, \theta_{2i})$ are obtained by

$$\begin{bmatrix} \theta_{oi} \\ \theta_{1i} \\ \theta_{2i} \end{bmatrix} = \begin{bmatrix} \tan^{-1}\left(\dfrac{y_{fi} - y_{si}}{x_{fi} - x_{si}}\right) \\ \tan^{-1}\left(\dfrac{d_{1i}}{\pm\sqrt{r_i^2 - d_{1i}^2}}\right) - \phi_i \\ \tan^{-1}\left(\dfrac{d_{2i}}{\pm\sqrt{r_i^2 - d_{2i}^2}}\right) - \phi_i \end{bmatrix} \tag{1}$$

where

$$r_i = \sqrt{\left(\frac{x_{fi} - x_{si}}{\cos\theta_{0i}} - l_0\right)^2 + z_i^2} \tag{2}$$

$$\phi_i = \tan^{-1}\frac{1}{z_i}\left(\frac{x_{fi} - x_{si}}{\cos\theta_{0i}} - l_0\right) \tag{3}$$

$$d_{1i} = \frac{1}{2l_1}\left\{\left(\frac{x_{fi} - x_{si}}{\cos\theta_{0i}} - l_0\right)^2 + z_i^2 + l_1^2 - l_2^2\right\} \tag{4}$$

$$d_{2i} = \frac{1}{2l_2}\left\{\left(\frac{x_{fi} - x_{si}}{\cos\theta_{0i}} - l_0\right)^2 + z_i^2 + l_1^2 + l_2^2\right\} \tag{5}$$

$$z_i = z_{fi} - z_{si} \tag{6}$$

Here, i denotes the leg number (i =1, 2, 3, 4). With reference to, l_0, l_1, and l_2 are lengths of links. The ultrasonic sensor and the force sensor are on the foot at l_3, where l_3 = 130 [mm].
Given the initial positions for soles and shoulders, joint angles of legs including the swing leg are obtained from Eq. (1). Note, however, that the operation of each joint is $90.0 \le \theta_{01} \le 245.0$, $-65.0 \le \theta_{02} \le 90.0$, $115.0 \le \theta_{03} \le 270.0$, $65.0 \le \theta_{04} \le -90.0$, $-65.0 \le \theta_{1i} \le 65.0$ and $-65.0 \le \theta_{2i} \le 90.0$ in degrees. Actions should be determined to not exceeding operational range.

3. Obstacle Detection and Recognition

To avoid an obstacle, its existence and height must be recognized using sensory information. For the robot, obstacles are detected and recognized by ultrasonic sensors. Ultrasonic sensors detect obstacles perpendicular to signals emitted by them. If the ultrasonic sensor was on the robot, the measurement of sensors would be shortened, so the ultrasonic sensors are on legs so that the emission of ultrasonic waves is changeable. As a result, obstacles not perpendicular to the robot's forward direction can be detected. When a leg is swinging, it can move up to a position where the ultrasonic sensor does not detect anything. Using this concept, the height of the obstacle is measured roughly.
Ultrasonic sensors on forelegs detect obstacles in the robot's forward direction. A downward sensor detects the unevenness of terrain. When the center of the robot's body is set as the origin, the position of an obstacle, (x_{ob}, y_{ob}), detected by the robot's front sensors is given by

$$x_{ob} = x_{si} + (l_0 + l_1 \cos\theta_{1i} + l_2 \sin\theta_{2i})\cos\theta_{0i} - L_{so} \sin\theta_{0i} \tag{7}$$
$$y_{ob} = y_{si} + (l_0 + l_1 \cos\theta_{1i} + l_2 \sin\theta_{2i})\sin\theta_{0i} + L_{so} \cos\theta_{0i} \tag{8}$$

The position of an obstacle detected by the robot's left and right sensors is given by

$$x_{ob} = x_{si} + (l_0 + l_1 \cos\theta_{1i} + l_2 \sin\theta_{2i})\cos\theta_{0i} + L_{so} \cos\theta_{0i} \tag{9}$$
$$y_{ob} = y_{si} + (l_0 + l_1 \cos\theta_{1i} + l_2 \sin\theta_{2i})\sin\theta_{0i} + L_{so} \sin\theta_{0i} \tag{10}$$

Here, i denotes the front leg numbers (i=1, 2). The position of the detected obstacle is calculated with the distance L_{so} measured by ultrasonic sensor, together with the body position and the joint angles of the foreleg.

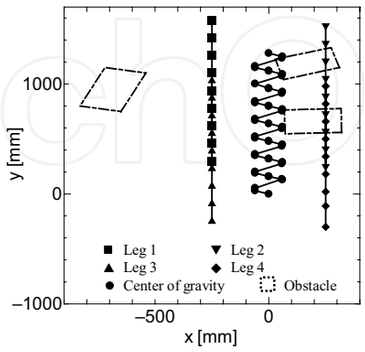

Fig. 4. Movement path of quadruped robot.

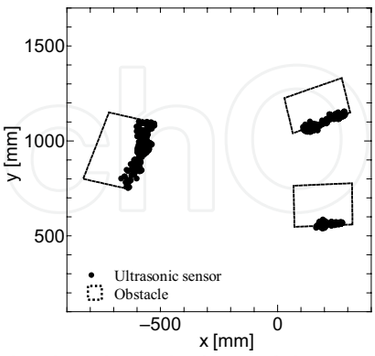

Fig. 5. Recognition result of the obstacles in the forward direction.

Positions of obstacles, detected where the robot moves (Fig. 4) are given in Figs. 5 and 6. Fig. 5 shows information regarding obstacles from foreleg sensors. We found that one side of each obstacle is detected. Fig. 6 shows positions and heights of obstacles when a leg passed over them, detected by the downward sensor. Robot action must be determined using such information.

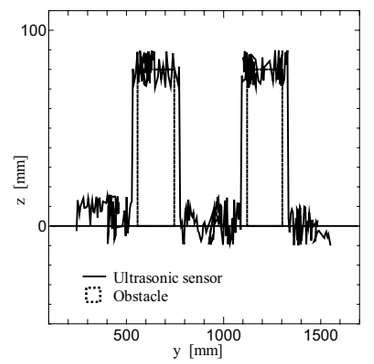

Fig. 6. Recognition result of the ground.

4. Action Determination of Quadruped Robot Using an NN

For an actual robot, information obtained from sensors includes leg joint angles and dimensions and heights of obstacles. We assumed that destination information is given in advance. Actions of the quadruped robot are determined by the above information. For each action, there are several combinatorial solutions such as the combination of forward- and turning-motions, one of forward- and sideway-motions, etc.

The robot action is decided by a three-layered NN (Fig. 7). This NN is trained offline so as to achieve an action using a minimum number of walking cycles.

Inputs to the NN are assumed to be the position of each sole $\{x_{f1}(k), y_{f1}(k)\}$, ···, $\{x_{f4}(k), y_{f4}(k)\}$, the robot's body height $Zr(k)$, x-directional maximum and minimum distances to an obstacle at right $\{x_{or\max}(k), x_{or\min}(k)\}$, the y-directional maximum and minimum distances to the obstacle at right $\{y_{or\max}(k), y_{or\min}(k)\}$, and the height of the obstacle at right $\{z_{or\max}(k), z_{or\min}(k)\}$, the x-directional maximum and minimum distances to the obstacle at left $\{x_{ol\max}(k), x_{ol\min}(k)\}$, the y-

directional maximum and minimum distances to the obstacle at left $\{y_{ol\max}(k), y_{ol\min}(k)\}$, and the height of the obstacle at left $\{z_{ol\max}(k), z_{ol\min}(k)\}$; x-directional distance error, i.e., distance between the COG of the robot and destination $x_{de}(k)$, y-directional distance error $y_{de}(k)$, and direction error, i.e., the direction between the destination and forward direction of the robot $\theta_{de}(k)$. Positions of each sole and obstacles are defined in the frame whose origin is fixed to the body center. Outputs of the NN are the amount of x- and y-directional movement of the robot $\{\Delta Xr(k), \Delta Yr(k)\}$ and the turning angle of the robot $\Delta\theta r(k)$. If no obstacles exist in front of the robot, measured values of obstacles are set to $(x_{or\max}, y_{or\max}, z_{or\max})=(1.0, -0.99, 0.0)$, $(x_{or\min}, y_{or\min}, z_{or\min})=(0.99, -1.0, 0.0)$, $(x_{ol\max}, y_{ol\max}, z_{ol\max})=(-0.99, -0.99, 0.0)$, and $(x_{ol\min}, y_{ol\min}, z_{ol\min})=(-1.0, -1.0, 0.0)$ [m], assuming the obstacle is behind the robot.

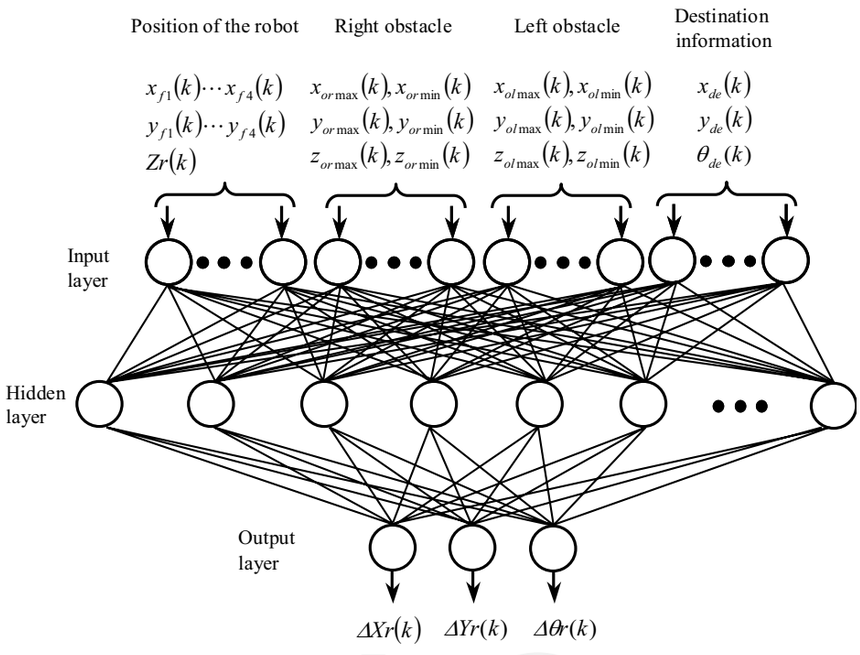

Fig. 7. Three layered neural network for determining the action of the quadruped robot.

A radial basis function neural network (RBFNN) (Elanayar & Shin, 1994, Sakawa & Tanaka, 1999), known as an NN that can realize various approximation functions, is used in the control system. With an RBFNN, a nonlinear function is expanded by any basis function having a circular contour, and is used as function approximation or pattern recognition. Unit functions at the hidden (or intermediate) layer of RBFNNs are given by

$$\phi_i(\boldsymbol{x}) = \exp\left\{-\frac{\left\|\boldsymbol{x}(k) - c_i\right\|^2}{\sigma_i^2}\right\} \tag{11}$$

where ϕ_i denotes ith unit output at the hidden layer, and design parameters of RBF are center $\boldsymbol{c}_i$ and standard deviation σ_i for each input. jth unit output at output layer o_j is given by

$$o_j(k) = \sum_{i=1}^{m} \omega_{ij} \phi_i(\boldsymbol{x}) \tag{12}$$

which is calculated by a linear combination of outputs of the hidden layer. Here, ω_{ij} denotes the connection weight between the ith hidden unit and the jth output unit, and m denotes the number of units at the hidden layer, where the number of hidden units is determined by trial and error. The number of hidden units was set to m=100, because a good result was obtained when m was four times the number of inputs.
Using this RBFNN, an action of the quadruped robot is determined from the information of the obstacles, the current state of the robot, and the destination information.

4.1 Acquisition of Obstacle Avoidance by Simulation

A block diagram of obstacle avoidance control system is shown in Fig. 8. To avoid obstacles, the system responds to the environment by altering the path and getting over or striding obstacles. To do so, the robot changes its height to that of an obstacle. Such actions constrain how much the robot can adjust the height. The RBFNN determines the movement of the robot using obstacle dimensions, the position of each leg, and robot height, collectively considered during walking. Positions of shoulders and legs are computed from the amount of movement. Each joint angle is calculated by Eq. (1), and the output is communicated to the robot.
Distance error is updated by change in the current COG position of the robot, after computing the COG position by considering positions of shoulders and legs. Although the position of an obstacle is measured by ultrasonic sensors of the robot, position information of obstacles in simulation is updated by change of the COG position of the robot. Each leg is changed based on the amount of robot movement. The range of leg placement is assumed as follows: when placing the leg on the ground, its range should be 50 [mm] away from a corner of an obstacle, whereas, when placing the leg on an obstacle, its range should be 30 [mm] away from a corner. If a leg cannot be placed in a position, the robot places it at the nearest position from the scheduled position.

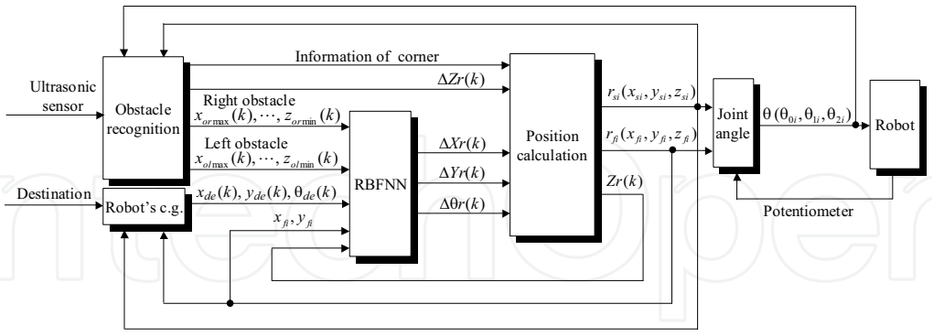

Fig. 8. Obstacle avoidance control system with consideration to the destination.

4.1.1 Simulation condition

The simulation environment is shown in Fig. 9. The y-axis is set to the forward direction of the robot. The robot is assumed to start from point (0.0, 0.0, 0.3) [m] and approach the goal, a circle having radius 0.2 [m], centered on the destination point.

Simulation is performed three times by setting distance error to (x_{de}, y_{de})=(0.0, 1.8), (0.15, 2.2), (–0.15, 2.2) [m]. We assume that one obstacle exists at the robot's right and the other at the left. Coordinates of legs and the obstacle are shown in Fig. 10. Initial positions for legs are set to (x_{f1}, y_{f1})=(–0.25, 0.3), (x_{f2}, y_{f2})=(0.25, 0.25), (x_{f3}, y_{f3})=(–0.25, –0.25), and (x_{f4}, y_{f4})=(0.25, –0.3) [m]. Positions of x- and y- coordinates of obstacles are shown in Table 1, where each row represents coordinate data combined to produce coordinates of any two obstacles. Height coordinates z_{or} and z_{ol} of obstacles are set to 0.06, 0.12, 0.3 [m], where the data can be combined to produce a combination of heights. Coordinate z_{or} (z_{ol})=0.3 [m] implies that it is an obstacle the robot cannot get over, so that the combination of z_{or}=0.3 [m] and z_{ol}=0.3 [m] is not considered in simulation. One of the two obstacles is assumed to be the obstacle that the robot cannot get over, so that simulation is conducted for 96 obstacles.

Walking is a crawl in which the body is supported by three or more legs. The order of swing leg selection is the right hind-leg, right foreleg, left hind-leg, and left foreleg. Realization of stable static walking is similar to the crawl. The COG of the robot should be in the polygon constructed by the supportive legs; therefore the body is moved so that the COG of the robot is always in the supportive leg polygon. We assume that the robot is parallel to the ground.

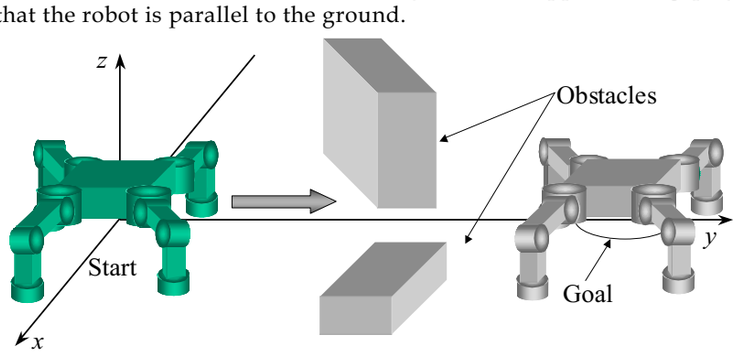

Fig. 9. The environmental setup for the acquisition of quadruped robot's action.

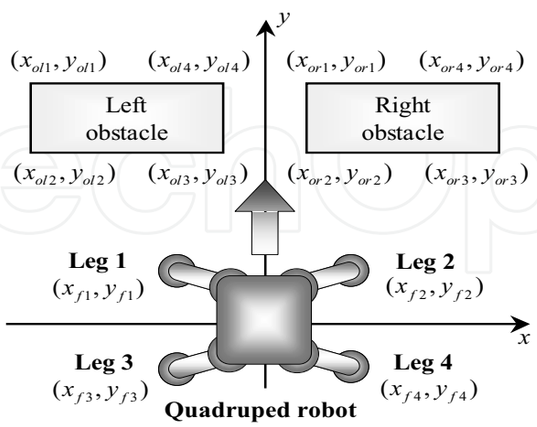

Fig. 10. Coordinates of the robot's legs and the obstacles.

No	x_{or1}	x_{or2}	x_{or3}	x_{or4}	x_{ol1}	x_{ol2}	x_{ol3}	x_{ol4}
1	0.5	0.1	0.1	0.5	− 0.5	− 0.1	− 0.1	− 0.5
2	0.5	0.1	0.1	0.5	− 0.5	−0.2	− 0.2	− 0.5
3	0.5	0.1	0.1	0.5	−0.5	− 0.1	− 0.1	− 0.5
4	0.5	0.1	0.1	0.5	− 0.5	− 0.2	− 0.2	− 0.5
No	y_{or1}	y_{or2}	y_{or3}	y_{or4}	y_{ol1}	y_{ol2}	y_{ol3}	y_{ol4}
1	0.85	0.65	0.55	0.75	0.86	0.66	0.65	0.85
2	0.85	0.65	0.66	0.86	0.75	0.55	0.65	0.85

Table 1. x- and y- directional coordinates of obstacles.

4.1.2 Setup of fitness function

In simulation, connection weights of the NN and parameters (center and standard deviations) of RBFs are optimized by a GA (Michalewicz, 1996) so that the robot avoids obstacles and reaches the destination with a minimum number of walk cycles. Table 2 shows design parameters for the GA used in simulation. The associated fitness function of an individual is defined by

$$fitness = \sum_{i=1}^{ob_n} (fitness_o + fitness_a + fitness_c) \tag{13}$$

whose solution is searched for as a minimization problem. ob_n is the number of environments considered in optimization. $fitness_o$ is an evaluation function associated with penalty for collision with an obstacle. $fitness_o$ is given by

$$fitness_o = \begin{cases} 0.0, \text{ if there is no collision} \\ [x_{de}^2(k) + y_{de}^2(k)] \times 10, \text{ otherwise} \end{cases} \tag{14}$$

Walking stops if the robot collides with an obstacle. $fitness_a$ is an evaluation function related to joint constraints, i.e., whether each joint angle is in an admissible range or not. $fitness_a$ is given by

$$fitness_a = \begin{cases} 0.0, \text{ if there is no outside of the range} \\ [x_{de}^2(k) + y_{de}^2(k)] \times 10, \text{ otherwise} \end{cases} \tag{15}$$

Walking stops if the joint exceeds joint constraints. $fitness_c$ is an evaluation function related to walk cycles required to reach the destination, and given by

$$fitness_c = [x_{de}^2(k) + y_{de}^2(k)] \times \frac{T}{50} \tag{16}$$

T denotes the walk cycles required to move from the starting point to the destination while avoiding obstacles by stable walking. The maximum number of walk cycles $T_{\max}$ in one environment is set to 50 and walking stops if the walk cycles exceed $T_{\max}$.

The number of individuals	100
Crossover rate	0.6 (uniform crossover)
Selection strategy	Tournament selection (3 individuals)
Elitist preserving strategy	10

Table 2. Design parameters for GA.

4.2 Simulation Result

4.2.1 Performance for training data

A typical control result after training the RBFNN using the above procedure is shown in Fig. 11. Environmental conditions for one of 96 combinations were as follows: x- and y-directional initial distance error of the destination were (x_{de}, y_{de})=(−0.15, 2.2) [m]; x- and y-directional coordinates of two obstacles were (x_{or1}, y_{or1})=(0.2, 0.85), (x_{or2}, y_{or2})=(0.2, 0.65), (x_{or3}, y_{or3})=(0.5, 0.55), (x_{or4}, y_{or4})=(0.5, 0.75), (x_{ol1}, y_{ol1})=(−0.5, 0.85), (x_{ol2}, y_{ol2})=(−0.5, 0.65), (x_{ol3}, y_{ol3})=(−0.1, 0.66), and (x_{ol4}, y_{ol4})=(−0.1, 0.86) [m]; and z-directional coordinates were z_{or}=0.3 and z_{ol}=0.12 [m].

The robot turned to the left to avoid an obstacle at right, in which it could not get over but conquered the obstacle at left. Legs 1 and 3 were used to get over the obstacle at left and the robot reached the destination. The number of walk cycles to the goal was 9 and the final distance error was (x_{de}, y_{de})=(–0.037, 0.046) [m].

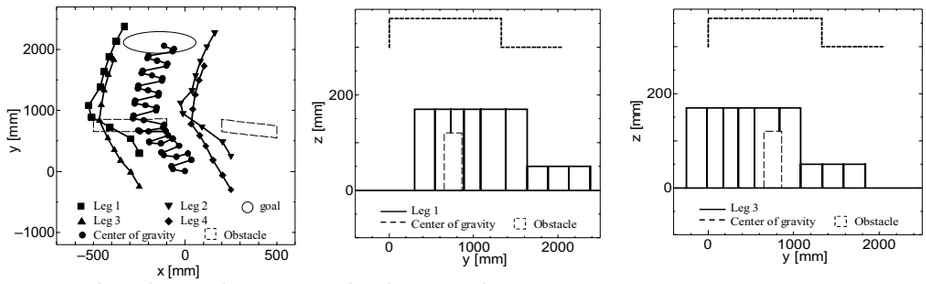

Fig. 11. Obstacle avoidance action for the trained environment.

4.2.2 Performance for untrained environment

In this section, we examine RBFNN performance in an untrained environment. We set the initial distance errors at (x_{de}, y_{de})=(0.35, 2.1) [m], and x- and y-directional coordinates of two obstacles at (x_{or1}, y_{or1})=(0.11, 0.87), (x_{or2}, y_{or2})=(0.11, 0.56), (x_{or3}, y_{or3})=(0.57, 0.55), (x_{or4}, y_{or4})=(0.57, 0.86), (x_{ol1}, y_{ol1})=(−0.56, 0.92), (x_{ol2}, y_{ol2})=(−0.56, 0.59), (x_{ol3}, y_{ol3})=(−0.14, 0.66), and (x_{ol4}, y_{ol4})=(−0.14, 0.99) [m], together with z-directional coordinates at z_{or}=0.07 and z_{ol} =0.3 [m].

Fig. 12 shows the results. In simulation, since the obstacle at left could not be gotten over by the robot, it moved to the right and got over it. We found that legs 2 and 4 were used for getting over the obstacle at right so that the robot reached the destination. The number of walk cycles to the goal was 9 and the final distance error was (x_{de}, y_{de})=(0.036, 0.104) [m].

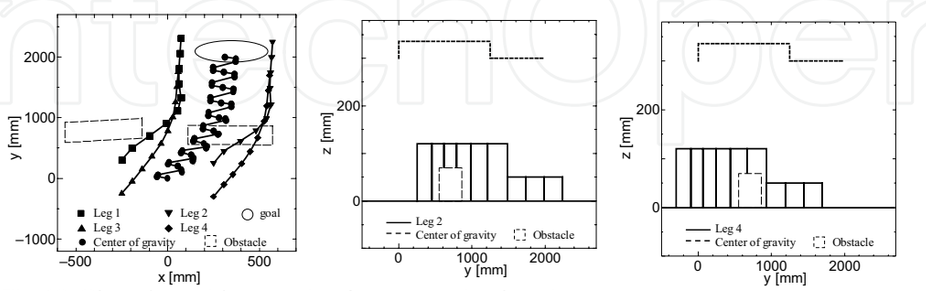

Fig. 12. Obstacle avoidance action for an untrained environment.

4.3 Experiments

Experiments were conducted by quadruped robot TITAN-VIII. Walk was at a crawl and the robot recognized an obstacle by ultrasonic sensors on the forelegs. x- and y-directional movement and the turning angle of the robot were determined by the NN from simulation. We assumed that the robot could not get over an obstacle taller than 0.15 [m]. Placement of a swing leg is decided by information retrieved by the downward ultrasonic sensor, depending on whether the placement is a corner of the obstacle. Whether the leg is a swing leg or not is decided by the force sensor.

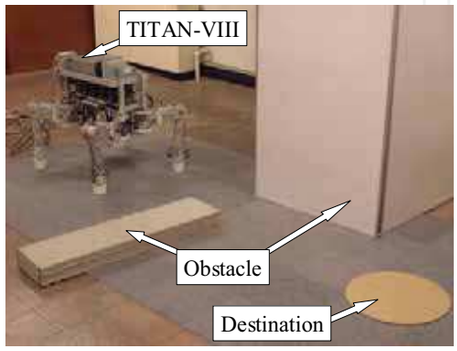

Fig. 13. Environment of an experiment.

The experimental environment is shown in Fig. 13. One obstacle is on the right and the other on the left. The obstacle at left can not be gotten over by the robot. Initial distance error was set to (x_{de}, y_{de})=(0.0, 1.9) [m].

Positions of obstacles are detected by the robot (Figs. 14 and 15). Fig. 14 shows information on obstacles gathered by sensors on the forelegs, whose results show that only one side of each obstacle could be detected. Fig. 15 shows positions and heights of obstacles when the leg has passed over them, which were detected by downward sensors. The path to the destination and the presence of obstacles is shown in Fig. 16. We found that the robot turned to the right to avoid the obstacle at left, which could not be gotten over, but got over the obstacle at right. Legs 2 and 4 were used for getting over the obstacle at right and the robot reached the destination. The number of walk cycles to the goal was 9 and the final distance error was (x_{de}, y_{de})=(0.137, 0.075) [m].

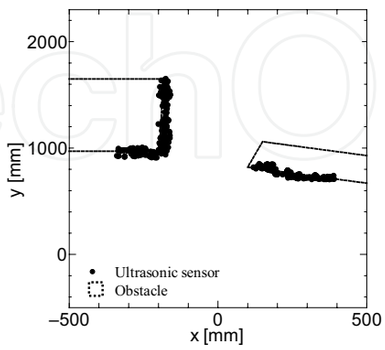

Fig. 14. Recognition result of the obstacles gathered by sensors on the forelegs.

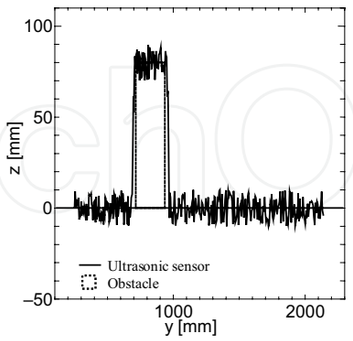

Fig. 15. Recognition result of the obstacles detected by downward sensors.

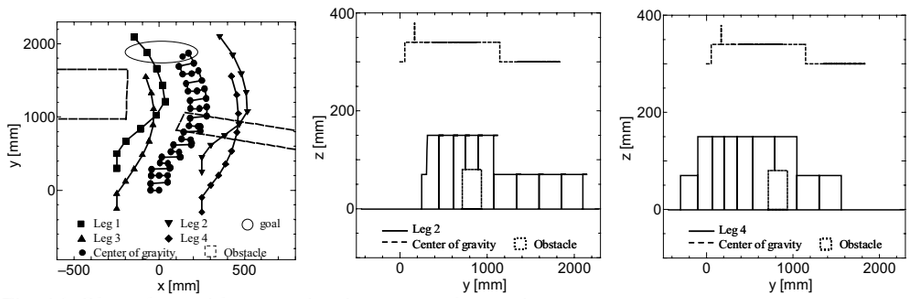

Fig. 16. Obstacle avoidance action in an actual experiment.

5. Determination of the Order of Swing Leg for Free-Gait

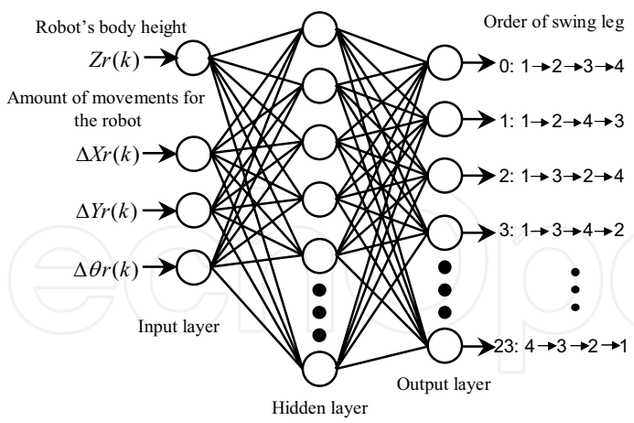

Fig. 17. Three layered neural network for determining the order of swing leg.

Since obstacle avoidance is taken into consideration, it is assumed that the static walk is adopted as a basic walk and the order of swing leg is determined. 24 kinds of the order exist in a static walk of the quadruped robot. Although 24 kinds of the order can be tried to implement whenever the

quadruped robot walks, in this research, the order of swing leg is determined by a three-layered NN shown in Fig. 17. Inputs to the NN are assumed to be the robot's body height $Zr(k)$, the amount of x- and y-directional movements of the robot $\{\Delta Xr(k), \Delta Yr(k)\}$ and the turning angle of the robot $\Delta\theta r(k)$. Moreover, we prepare 24 units at the output, corresponding to 24 kinds of the order (Table 3). The order of swing leg fed to the quadruped robot uses the order associated to the unit whose output value is closest to one among output units. RBFNN is also used for the NN, where the number of units in the hidden layer is set to 50.

Output number	Order of swing leg	Output number	Order of swing leg
0	1→2→3→4	12	3→1→2→4
1	1→2→4→3	13	3→1→4→2
2	1→3→2→4	14	3→2→1→4
3	1→3→4→2	15	3→2→4→1
4	1→4→2→3	16	3→4→1→2
5	1→4→3→2	17	3→4→2→1
6	2→1→3→4	18	4→1→2→3
7	2→1→4→3	19	4→1→3→2
8	2→3→1→4	20	4→2→1→3
9	2→3→4→1	21	4→2→3→1
10	2→4→1→3	22	4→3→1→2
11	2→4→3→1	23	4→3→2→1

Table 3. Order of swing leg to each output of NN.

5.1 Acquisition of the Order of Swing Leg

In a static walk of quadruped robot, since the amount of movements of the body changes with the order of swing leg, the robot produces different movable range for each order of swing leg. For this reason, if the stability of static walk is maintained by less movement of the body, then the movable range of each leg becomes large; it can imsequently enlarge the movable range of the robot.

For teacher signal used for this research, when the robot's body height $Zr(k)$ was changed from 300 [mm] to 370 [mm], the amount of x-directional movement of the robot $\Delta Xr(k)$ is changed from −100 [mm] to 100 [mm], the amount of y-directional movement of the robot $\Delta Yr(k)$ is changed from 150 [mm] to 350 [mm] and the turning angle of the robot $\Delta\theta r(k)$ is changed form −15 [degree] to 15 [degree], respectively, the order of swing leg that the movement of the robot's body is a minimum and the stability of static walk is satisfied is set as one, and the other order is set to zero. Here, there were 19 kinds of the order of swing leg that the amount of movements of the robot's body became the minimum. Note however that when the amount of changes of $Zr(k)$, $\Delta Xr(k)$, $\Delta Yr(k)$, and $\Delta\theta r(k)$ is fixed, the number of selections is changed, depending on the order of swing leg. Therefore, the amount of change of $Zr(k)$, $\Delta Xr(k)$, $\Delta Yr(k)$, and $\Delta\theta r(k)$ is enlarged for the case of high number of selections, whereas the amount of change of movement is made small for the case of low number of selections, and 20 data are prepared for each order of swing leg. Moreover, it is assumed that initial leg positions of quadruped robot are set to $(x_{f1}, y_{f1})=(-0.25, 0.3)$, $(x_{f2}, y_{f2})=(0.25, 0.25)$, $(x_{f3}, y_{f3})=(-0.25, -0.25)$, and $(x_{f4}, y_{f4})=(0.25, -0.3)$ [m]. Here, the subscript number denotes the leg number.

In this research, the order of swing leg is determined using RBFNN. Connection weights of the NN and parameters (center and standard deviations) of RBFs are optimized by a GA so that the relation between an input and an output is satisfied. There is 19 kinds of the order of swing leg chosen when changing $Zr(k)$, $\Delta Xr(k)$, $\Delta Yr(k)$, and $\Delta\theta r(k)$, respectively, and 20 data are prepared for each one. Here, there are a total of 380 kinds of combination optimized by using GA. The associated fitness function of an individual is defined by

$$fitness = \sum_{i=1}^{n} (fitness_A + fitneess_B) \tag{17}$$

whose solution is searched for as a minimization problem. n is the total number of combinations in optimization. $fitness_A$ is an evaluation function only applied when a teacher signal ts_j is one, which is given by

$$fitness_A = \sqrt{(o_j - ts_j)^2}\Big|_{j=j1} \tag{18}$$

where j denotes any unit number of output layer and $j1$ denotes the unit number of $ts_j \equiv 1$. Contrarily, $fitness_B$ is an evaluation function applied when a ts_j is zero, which is given by

$$fitness_B = \sqrt{\max\{(o_1 - ts_1)^2, \cdots, (o_j - ts_j)^2, \cdots, (o_{24} - ts_{24})^2\}} \tag{19}$$

$fitness_B$ denotes the largest value in the difference of o_j and ts_j. j denotes the unit number 1 to 24 except for the case of $j1$ that denotes the unit number of $ts_j \equiv 1$.

5.2 Obstacle Avoidance with Consideration to the Free-Gait

A block diagram of obstacle avoidance control system considered here is shown in Fig. 18. The avoidance action of quadruped robot is determined by the upper NN. Furthermore, the order of swing leg is determined by the lower NN from the amount of movements of the robot.

The simulation environment is the same as section 4.1. The y-axis is set to the forward direction of the robot. The robot is assumed to start from point (0.0, 0.0, 0.3) [m] and approach the goal, a circle having radius 0.2 [m], centered at the destination point.

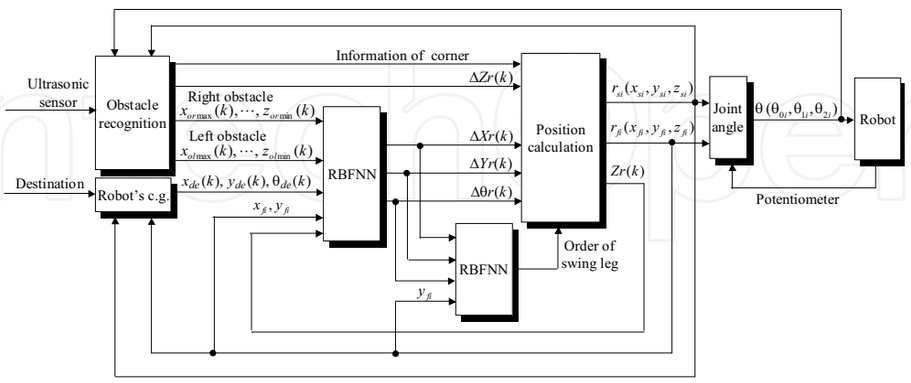

Fig. 18. Obstacle avoidance control system with consideration to the order of swing leg.

We set the initial distance errors at (x_{de}, y_{de})=(− 0.15, 2.2) [m], and x- and y-directional coordinates of two obstacles at (x_{or1}, y_{or1})=(0.1, 0.85), (x_{or2}, y_{or2})=(0.1, 0.65), (x_{or3}, y_{or3})=(0.5, 0.66), (x_{or4}, y_{or4})=(0.5, 0.86), (x_{ol1}, y_{ol1})=(−0.5, 0.75), (x_{ol2}, y_{ol2})=(−0.5, 0.55), (x_{ol3}, y_{ol3})=(−0.2, 0.65), and (x_{ol4}, y_{ol4})=(−0.2, 0.85) [m], together with z-directional coordinates at z_{or}=0.12 and z_{ol}=0.3 [m]. Here, although a robot can get over an obstacle at right, an obstacle at left shall not be get over. The robot's body height $Zr(k)$ is adjusted and changed to the height of the obstacle which can be get over.

5.2.1 When the order of swing leg is fixed

The movement path of the quadruped robot when fixing the order of swing leg and avoiding an obstacle is shown in Fig. 19, and the corresponding amount of movements of the robot's body Br is shown in Table 4. Here, the leg number used as swing leg is assigned as the left foreleg to 1, right foreleg to 2, left hind-leg to 3 and right hind-leg to 4, as shown in Fig. 2. The robot's body height changes as shown in Fig. 20. Furthermore, it can be checked from Fig. 20 that both the leg 2 and 4 were used for a getting over to the obstacle at right.

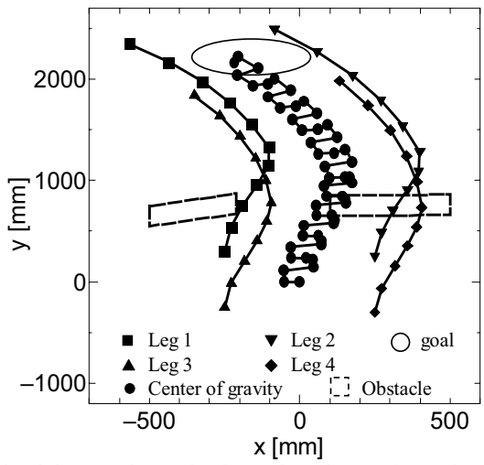

Fig. 19. Movement path of the quadruped robot when fixing the order of swing leg.

$\Delta Xr(k)$ [mm]	$\Delta Yr(k)$ [mm]	$\Delta\theta r(k)$ [deg]	$Zr(k)$ [mm]	Br [mm]	Order of swing leg
22.026	237.002	0.809	300.0	418.090	4→2→3→1
43.958	217.174	−0.818	360.0	372.904	4→2→3→1
45.347	200.484	−0.639	360.0	354.896	4→2→3→1
32.227	189.676	1.413	360.0	362.871	4→2→3→1
12.261	186.691	4.182	360.0	390.217	4→2→3→1
−15.944	241.234	3.894	360.0	520.653	4→2→3→1
−17.443	240.270	4.289	360.0	528.034	4→2→3→1
−18.958	239.279	4.695	300.0	535.676	4→2→3→1
−20.522	238.245	5.122	300.0	543.786	4→2→3→1
−22.187	237.142	5.586	300.0	552.667	4→2→3→1

Table 4. The amount of movements of the robot's body when fixing the order of swing leg.

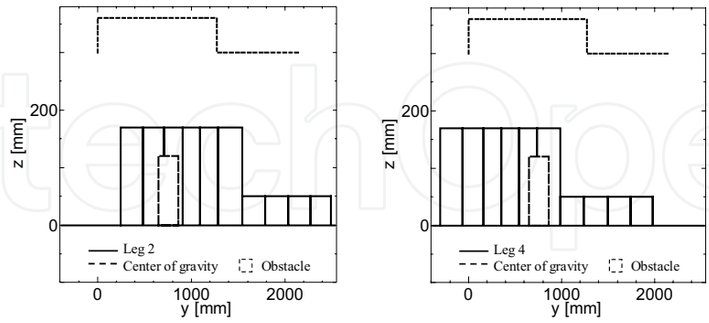

Fig. 20. Movement path of the leg 2 and 4.

5.2.2 When all 24 kinds of the order are tried

The movement path of the quadruped robot, when trying all 24 kinds of the order and avoiding an obstacle, is shown in Fig. 21, and the amount of movements of the robot's body *Br* is shown in Table 5. Here, the robot's body height changes with obstacles. For this reason, the robot's body height varies as shown in Fig. 20. Compared to the case where the order of swing leg is fixed, it is found that the amount of movements of the robot's body is relatively small.

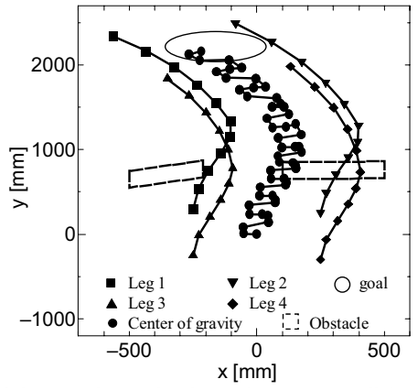

Fig. 21. Movement path of the quadruped robot when trying all 24 kinds of the order.

$\Delta Xr(k)$ [mm]	$\Delta Yr(k)$ [mm]	$\Delta\theta r(k)$ [deg]	$Zr(k)$ [mm]	Br [mm]	Order of swing leg
22.026	237.002	0.809	300.0	417.078	2→4→3→1
43.958	217.174	−0.818	360.0	372.904	4→2→3→1
45.347	200.484	−0.639	360.0	354.896	4→2→3→1
32.227	189.676	1.413	360.0	362.871	4→2→3→1
12.261	186.691	4.182	360.0	389.931	2→4→3→1
−15.944	241.234	3.894	360.0	520.653	4→2→3→1
−17.443	240.270	4.289	360.0	528.034	4→2→3→1
−18.958	239.279	4.695	300.0	448.019	3→1→4→2
−20.522	238.245	5.122	300.0	446.065	3→1→4→2
−22.187	237.142	5.586	300.0	443.984	3→1→4→2

Table 5. The amount of movements of the robot's body when trying all 24 kinds of the order.

5.2.3 When the order of swing leg is determined by RBFNN

After training the lower RBFNN, the movement path of the quadruped robot, determining the order of swing leg and avoiding an obstacle, is shown in Fig. 22. The corresponding amount of movements of the robot's body *Br* is shown in Table 6. Note here that the robot's body height is the same as that shown in Fig. 20. Compared to the case where the order of swing leg is fixed, it is observed that the amount of movements of the robot's body is small. However, compared with the case where all 24 kinds of the order are tried, the amount of movements of t he body was slightly large.

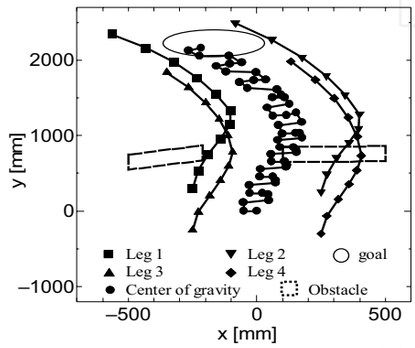

Fig. 22. Movement path of the quadruped robot when determining the order of swing leg using RBFNN.

$\Delta Xr(k)$ [mm]	$\Delta Yr(k)$ [mm]	$\Delta\theta r(k)$ [deg]	$Zr(k)$ [mm]	Br [mm]	Order of swing leg
22.026	237.002	0.809	300.0	418.090	4→2→3→1
43.958	217.174	−0.818	360.0	372.904	4→2→3→1
45.347	200.484	−0.639	360.0	354.896	4→2→3→1
32.227	189.676	1.413	360.0	362.871	4→2→3→1
12.261	186.691	4.182	360.0	390.217	4→2→3→1
−15.944	241.234	3.894	360.0	520.653	4→2→3→1
−17.443	240.270	4.289	360.0	528.034	4→2→3→1
−18.958	239.279	4.695	300.0	448.019	3→1→4→2
−20.522	238.245	5.122	300.0	446.065	3→1→4→2
−22.187	237.142	5.586	300.0	443.984	3→1→4→2

Table 6. The amount of movements of the robot's body when determining the order of swing leg using RBFNN.

6. Re-learning of the NN for Determining the Robot Action

NN for determining the robot action is acquired by re-learning the NN that was built in the case when the order of swing leg was fixed.

A block diagram of obstacle avoidance control system is the same as section 5.2. Moreover, the simulation condition and fitness function are the same as section 4.1.

6.1 When the Obstacle at Right is a Wall

We set the initial distance errors at (x_{de}, y_{de})=(−0.15, 2.2) [m], and x- and y-directional coordinates of two obstacles at (x_{or1}, y_{or1})=(0.2, 0.85), (x_{or2}, y_{or2})=(0.2, 0.65), (x_{or3}, y_{or3})=(0.5, 0.55), (x_{or4}, y_{or4})=(0.5,

0.75), (x_{ol1}, y_{ol1})=(−0.5, 0.85), (x_{ol2}, y_{ol2})=(−0.5, 0.65), (x_{ol3}, y_{ol3})=(−0.1, 0.66), and (x_{ol4}, y_{ol4})=(− 0.1, 0.86) [m], together with z-directional coordinates at z_{or}=0.3 and z_{ol}=0.12 [m]. Here, although a robot can get over an obstacle at left, an obstacle at right shall not be get over.

The simulation result using the learned RBFNN is shown in Fig. 23. The number of walk cycles to the goal was 10, and the distance from the COG of the robot to the destination point was (x_{de}, y_{de})=(0.017, −0.027) [m]. Moreover, the amount of movements of the robot and the order of swing leg are shown in Table 7. It was found that the order of the swing leg changes with the amount of movements of the robot.

In the simulation result of the case where unlearned RBFNN is used, the number of walk cycles to the goal was 9, and the distance from the COG of the robot to the destination point was (x_{de}, y_{de})=(−0.037, −0.046) [m].

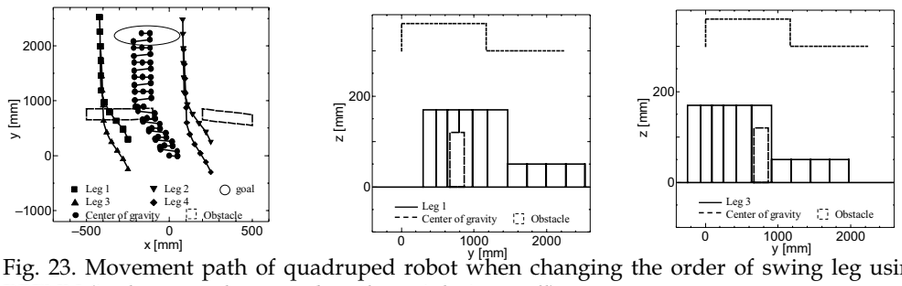

Fig. 23. Movement path of quadruped robot when changing the order of swing leg using RBFNN (in the case where an obstacle at right is a wall).

$\Delta Xr(k)$ [mm]	$\Delta Yr(k)$ [mm]	$\Delta\theta r(k)$ [deg]	$Zr(k)$ [mm]	Order of swing leg
−26.645	183.469	0.751	300.0	3→1→4→2
−35.059	157.334	0.497	360.0	3→1→4→2
−32.240	162.725	0.311	360.0	3→1→4→2
−24.792	181.329	0.089	360.0	3→1→4→2
−14.077	211.723	−0.166	360.0	3→1→4→2
2.710	274.449	−0.255	360.0	4→2→3→1
2.834	266.891	−0.172	300.0	4→2→3→1
1.787	265.774	−0.159	300.0	4→2→3→1
0.661	264.499	−0.143	300.0	4→2→3→1
−0.516	263.101	−0.126	300.0	4→2→3→1

Table 7. The amount of movements of the robot in the case where an obstacle at right is a wall.

6.2 When the Obstacle at Left is a Wall

We set the initial distance errors at (x_{de}, y_{de})=(−0.15, 2.2) [m], and x- and y-directional coordinates of two obstacles at (x_{or1}, y_{or1})=(0.1, 0.85), (x_{or2}, y_{or2})=(0.1, 0.65), (x_{or3}, y_{or3})=(0.5, 0.66), (x_{or4}, y_{or4})=(0.5, 0.86), (x_{ol1}, y_{ol1})=(−0.5, 0.75), (x_{ol2}, y_{ol2})=(−0.5, 0.55), (x_{ol3}, y_{ol3})=(−0.2, 0.65), and (x_{ol4}, y_{ol4})=(−0.2, 0.85) [m], together with z-directional coordinates at z_{or}=0.12 and z_{ol}=0.3 [m]. Here, although a robot can get over an obstacle at right, an obstacle at left shall not be get over.

The simulation result is shown in Fig. 24. The number of walk cycles to the goal was 11, and the distance from the COG of the robot to the destination point was (x_{de}, y_{de})=(0.018, −0.014) [m].

Moreover, the amount of movements of the robot and the order of swing leg are shown in Table 8. In the simulation result when unlearned RBFNN is used, the number of walk cycles to the goal was 10, and the distance from the COG of the robot to the destination point was (x_{de}, y_{de})=(0.068, 0.027) [m].

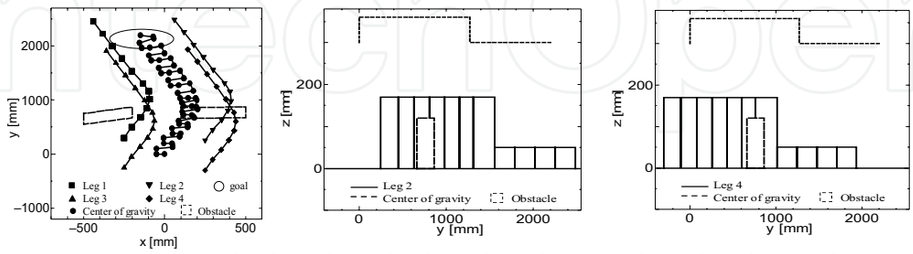

Fig. 24. Movement path of quadruped robot when changing the order of swing leg using RBFNN (in the case where an obstacle at left is a wall).

$\Delta Xr(k)$ [mm]	$\Delta Yr(k)$ [mm]	$\Delta\theta r(k)$ [deg]	$Zr(k)$ [mm]	Order of swing leg
45.379	198.899	−0.298	300.0	4→2→3→1
55.253	181.795	0.837	360.0	4→2→3→1
49.297	172.875	1.419	360.0	4→2→3→1
36.077	165.824	1.896	360.0	2→4→3→1
18.925	159.794	2.220	360.0	2→4→3→1
−0.386	155.208	2.319	360.0	2→4→1→3
−23.440	241.344	0.167	360.0	4→2→3→1
−23.620	239.286	0.087	300.0	4→2→3→1
−24.022	237.751	0.013	300.0	4→2→3→1
−24.397	236.259	−0.060	300.0	4→2→3→1
−24.746	234.816	−0.131	300.0	4→2→3→1

Table 8. The amount of movements of the robot in the case where an obstacle at left is a wall.

7. Conclusions

We experimentally have proved a method for acquiring a path to a destination and obstacle avoidance of a quadruped robot. Robot actions were determined through an RBFNN, whose input consisted of destination information, obstacle configuration, and current robot status. Using training data on environmental conditions, focusing on x-, y-, and z-coordinates of different obstacles and certain destinations, RBFNN design parameters were optimized using a GA so that the robot reached the destination with a minimum number of walking cycles. For an untrained (unknown) environment, we found that the RBFNN was useful for acquiring an obstacle avoidance path to the destination. Effectiveness of this approach was examined by actual experiments. However, free-gait motion was not taken into consideration in the first reseach.

A method of determining the order of swing leg in free gait by an RBFNN, whose inputs are the amount of movements for the quadruped robot and the height of the body, has been proposed for the second research. In the tuning of design parameters of the RBFNN, 20 data to which the amount of movements for the robot was changed are prepared for each order of swing leg. Such design parameters were optimized using GA so that the relation between an input and an output is satisfied.

As a result, compared to the case where the order of swing leg is fixed, the amount of movements for the robot body was small. However, compared to the case where all 24 kinds of the order are tried, that for the robot body was slightly large. It seems to be attributed to the fact that there was few data used for the study of the RBFNN. In order to make the amount of movements for the robot body much smaller, more data need to be used for the study of RBFNN.

In order to acquire the obstacle avoidance action of quadruped robots with considering to the order of swing leg, the action of a quadruped robot has been determined through an RBFNN, whose inputs were the destination information, the obstacle configurations, and the robot's self-state. The NN for determining the robot's action is acquired by re-learning the NN that was built in the case where the order of swing leg was fixed. Compared to the case where the unlearned RBFNN is used, the final distance error to the destination of the present approach was small; however the walk cycle was comparable to each other. It is attributed to the fact that a priority was assigned to the error distance in the evaluation of GA. For this reason, in order to make a walk cycle smaller, further fitness function of GA needs to be re-examined. Moreover, the effectiveness of the proposed system needs to be verified by using the actual system.

Obstacles are recognized by an ultrasonic sensor that detects reflected ultrasonic waves on a flat surface. Obstacles were assumed to be flat, i.e., rectangular blocks and oblique obstacles were not considered. Since the order of the swing leg was assumed to be constant, this assumption appears to have slightly restricted the robot action. We will improve the mobility efficiency of the robot by constructing a system that changes the sequence of the swing leg based on environmental conditions and improve recognition system by adding a vision sensor (Chow & Chung, 2002).

8. References

Chen, X.; Watanabe, K.; Kiguchi, K. & Izumi, K. (2002). An ART-based fuzzy controller for the adaptive navigation of a human-coexistent quadruped robot, *IEEE/ASME Transactions on Mechatronic*s, Vol. 7, No 3, pp. 318-328

Chen, X.; Watanabe, K.; Kiguchi, K. & Izumi, K. (2002). Path tracking based on closed-loop control for a quadruped robot in a cluttered environment, *ASME, Journal of Dynamic Systems, Measurement, and Control*, Vol. 124, pp. 272-280

Chow, Y.H. and Chung, R. (2002), VisionBug: A hexapod robot controlled by stereo cameras, *Autonomous Robots*, Vol. 13, No. 3, pp. 259-276.

Elanayar, V. T. S. & Shin, Y. C. (1994). Radial basis function neural network for approximation and estimation of nonlinear stochastic, *IEEE Transactions on Neural Networks*, Vol. 5, No. 4, pp. 594-603

Furusho, J. (1993). Research deployment of the walking robot, *Journal of Robotics Society of Japan*, Vol. 11, No 3, pp. 306-313

Hirose, S. & Arikawa, K. (1999). Development of quadruped walking robot TITAN-VIII for commercially available research platform, *Journal of Robotics Society of Japan*, Vol. 17, No 8, pp. 1191-1197

Hirose, S. & Yoneda, K. (1993). Toward development of practical quadruped walking vehicle, *Journal of Robotics Society of Japan*, Vol. 11, No 3, pp. 360-365

Kimura, H. (1993). Dynamic walk of the quadruped robot, *Journal of Robotics Society of Japan*, Vol. 11, No 3, pp. 372-378

Kimura, H.; Shimoyama, I. & Miura, H. (1990). Dynamics in the dynamic walk of a quadruped robot, *Advanced Robotics*, Vol. 4, No. 3, pp. 283-301

Michalewicz, Z. (1996). G*enetic Algorithms* + *Data Structure* = *Evolution Programs*, 3rd, revised and extended edition ed., Springer, Germany

Nakamura, T.; Seki, M.; Mori, Y. & Adachi, H. (1999). Free gait planning using Monte Carlo method for locomotion on rugged terrain, *1999 JSME Conference on Robotics and Mechatronics*, 1A1-42-061 (CD-ROM)

Şafak, K. K. & Adams, G. G. (2002). Dynamic modeling and hydrodynamic performance of biomimetic underwater robot locomotion, *Autonomous Robots*, Vol. 13, No. 3, pp. 223-240

Sakawa, M. & Tanaka, M. (1999). *Introduction to Neurocomputing*, Tokyo, Morikita Shuppan Co., Ltd.

Mobile Robotics, Moving Intelligence

Edited by Jonas Buchli

ISBN 3-86611-284-X

Hard cover, 586 pages

Publisher Pro Literatur Verlag, Germany / ARS, Austria

Published online 01, December, 2006

Published in print edition December, 2006

This book covers many aspects of the exciting research in mobile robotics. It deals with different aspects of the control problem, especially also under uncertainty and faults. Mechanical design issues are discussed along with new sensor and actuator concepts. Games like soccer are a good example which comprise many of the aforementioned challenges in a single comprehensive and in the same time entertaining framework. Thus, the book comprises contributions dealing with aspects of the Robotcup competition. The reader will get a feel how the problems cover virtually all engineering disciplines ranging from theoretical research to very application specific work. In addition interesting problems for physics and mathematics arises out of such research. We hope this book will be an inspiring source of knowledge and ideas, stimulating further research in this exciting field. The promises and possible benefits of such efforts are manifold, they range from new transportation systems, intelligent cars to flexible assistants in factories and construction sites, over service robot which assist and support us in daily live, all the way to the possibility for efficient help for impaired and advances in prosthetics.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen